

# GE423 Laboratory Assignment 7

## Robot Actuators and Servos

**Recommended Due Date: By your lab time the week of April 6<sup>th</sup>**  
**Possible Points: If checked off before your lab time the week of Apr. 13<sup>th</sup> ... 14 points**  
**If checked off after your lab time the week of Apr. 13<sup>th</sup> and before 4:00PM Apr. 17<sup>th</sup> ... 11 points**  
**If work not finished before 4:00PM Apr. 17<sup>th</sup> ... 0 points**

### Goals for this Lab Assignment:

1. Use a new Project Creator that includes interfaces to all needed sensors for the final project (besides the LADAR).  
The new project creator is located at  
**V:\c6713dsk\project\_creator\ge423\segmentation\Segmentation\_Project\_Creator.exe.**
2. Learn about hobby servos for simple actuation.
3. Learn how to add feeler switches to the robot.
4. Use another new Project Creator that adds the laser range finder's (LADAR) code to the vision segmentation code. This new project creator is located at  
**V:\c6713dsk\project\_creator\ge423\segmentation\_and\_ladar\Ladar\_and\_Segmentation\_Project\_Creator.exe.**
5. Modify the robot to grab objects and move them to specific locations.

### DSP/BIOS Objects Used:

Any you wish

### Library Functions Used:

Init\_RCservo, out\_RCservo, read\_parallel\_data,

**Prelab:** (none)

### Laboratory Exercise

#### Exercise 1: Create a New Project

For Labs 7, 8 and the Final Project you will use a new project creator that includes code that interfaces with most of the sensors and actuators on the robot. This first exercise introduces you to the new project creator and its layout. It is located at the directory: **V:\c6713dsk\project\_creator\ge423\segmentation\Segmentation\_Project\_Creator.exe.**

1. Look through all the `user_*.c` files to find all the code that has been given to you and note where each sensor's code is located.
2. The default code for this new project creator allows you to check that the robot's sensors and actuators are working correctly. By changing the state of the dipswitches you can test different sensors.

Dipswitch Setting	Function	✓ Check
Case 0: (all switches down)	3 IR Distance Sensors Connected to DSP Board	
Case 1:	2 Ultrasonic Distance Sensor Readings	

Case 2:	2 Photo-resistor readings from Ultrasonic Sensors	
Case 3:	5 IR Distance Sensors Connected to Atmel Board & 4 analog IR Distance Sensors Connected to Atmel Board	Ask TA to connect Sensors
Case 4:	4 Optical Encoders, Left, Right, Encoder 3 & 4	
Case 5:	Gyro Angle and Gyro Rate	
Case 6:	ADC Channels 1, 2 and 4. (ADC3 is Gyro above) ADC4 is Gyro's temperature sensor.	Ask TA on how to increase temperature reading.
Case 7:	Compass Angle in Radians and Raw Compass Readings (See datasheet for Raw reading explanation).	
Case 8:	Print the RGB value and HSV value of the pixel selected by the blue cross hair. Blue Cross Hair moved by Mouse.	Ask TA to Connect Mouse
Case 9:	Send vision data over Serial port 2 to PC.	We will do this in Lab 8
Case 10:	Echo 4 ADC readings directly to 4 DAC outputs	Don't need to test this one.
Case 11:	X and Y pixel location of the largest bright red/orange object. Green Cross Hair indicates centroid of this object.	Ask TA for an orange object.
Default	Undefined	—

### Exercise 2: Get 2 RC servos working using DSP PWM output. (82C54 Timer Chip)

There are two functions `void Init_RCservo(void)` and `void out_RCservo(int channel, float u)` that you will use to drive the RC servos (See `RCservo.c`). Note in `out_RCservo` that `u` is in units of milliseconds. It is limited to a range from 0.6 to 3.0. `Init_RCservo` is a function you will call in `main` to initialize the daughter card to command the RC servos. (It is already called by default.) Find the limits of your RC servos by modifying the default code to read the value of encoder 3 and send its value divided by 100 to the `out_RCservo` function. Print this value to the LCD screen every 100ms and determine your limits. The default code steps the RC servos back and forth between set positions.

### Exercise 3: Get 2 RC servos working using Atmel PWM output.

As in exercise 2, find the limits of your RC servos when connected to the Atmel microcontroller. With the Atmel the possible values sent to the RC servos are integers in the range 0 to 255. Again use encoder 3 to adjust the values sent to the Atmel and print the value to the LCD screen. The project creator generates the example code below automatically. This code steps the RC servos back and forth between two positions. For this exercise you will modify the default code. To send a value to the Atmel, assign new values to `Atmel_RC1` and `Atmel_RC2` and post the semaphore `SEM_atmel_rcservo`.

```

if (0 == (timeint%3000)) {
    if (stepRCservo == 1) {
        Atmel_RC1 = 75;
        Atmel_RC2 = 75;
        Atmel_RC3 = 75;
        SEM_post(&SEM_atmel_rcservo);

        out_RCservo(1,1.2);
        out_RCservo(2,1.2);
        stepRCservo = 0;
    } else {
        Atmel_RC1 = 175;
        Atmel_RC2 = 175;
        Atmel_RC3 = 175;
        SEM_post(&SEM_atmel_rcservo);

        out_RCservo(1,2.0);

```

```
        out_RCservo(2,2.0);  
        stepRCservo = 1;  
    }  
}
```

#### Exercise 4: Getting the Feeler Switches working

We are going to connect two switches to the parallel port connector on the DSK's daughter board. To read the state of the switches (ON or OFF), use the function `int read_parallel_data(void)`. Parallel port bits 0 and 1 will be connected to the two switches. The function `read_parallel_data` returns an 8-bit number that indicates the state of parallel port input bits 0-7. For the two feeler switches, you will be interested in the least significant two bits. With the power OFF on your robot, hookup two switches to your robot and write code to make sure they are working printing the value returned by `read_parallel_data` to the LCD screen.

#### Exercise 5: Trying out the Laser Range Finder (LADAR) sensor. LADAR stands for LAsER Detection And Ranging.

For this exercise and for any project using the LADAR you will need to use yet another project creator. It is located at the directory:

V:\c6713dsk\project\_creator\ge423\segmentation\_and\_ladar\Ladar\_and\_Segmentation\_Project\_Creator.exe. It has all the code given in the previous project creator and adds code for the LADAR.

1. The DSP communicates to the LADAR through the second RS-232 serial port on the robot (UART2). Look through the UART2 receive task in `user_UARTFuncs.c` to locate where you will add your code to use the LADAR data for the final project. For this exercise you will only be viewing the data on the Robot's color LCD and in MATLAB.
2. Build and run your new project's code. Set the DSK's jumpers to 12 (0 and 1 down, 2 and 3 up). After 6 seconds of zeroing the gyro sensor, the LADAR data will be displayed on the robot's color LCD screen. Rotate the robot and put obstacles in the LADAR's path to get an understanding of the data received from the LADAR.
3. By default, the LADAR data seen on the robot's color LCD is displayed in the robot's coordinate frame. So (0,0) is the center of the robot. You can also change the LCD to display LADAR readings in "World" coordinates. See Figure 1 for the definition of the World coordinate system. In `user_colorVisionFuncs.c` scroll down towards the end of the function `userProcessColorImageFunc_laser` and find the function call to `UpdateLCDwithLADAR(ptrImage,0)`; The second parameter can either be 0 (robot coordinate frame) or 1 (World coordinate frame). Change the 0 to a 1 and rebuild and run your project. Then when your project is running, add the structure variable `ROBOTps` to Code Composer's "Watch Window." Change the value of `ROBOTps.x`, `ROBOTps.y` and `ROBOTps.theta` and notice what happens to the location of the robot and its LADAR data on the LCD screen. In your final project you will be updating this `ROBOTps` structure with the actual location of the robot.

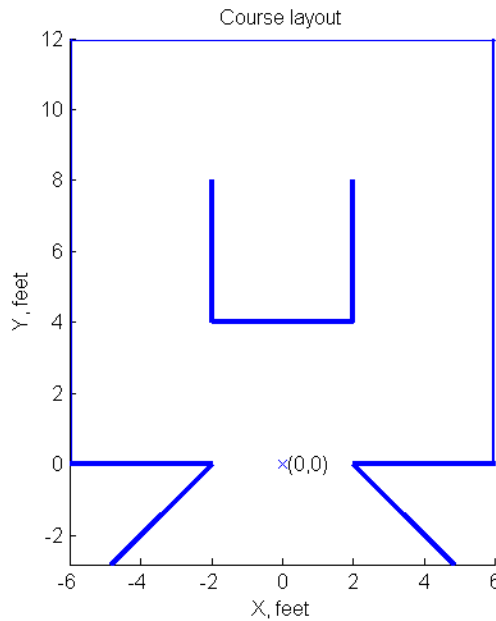


Figure 1: World Coordinate System

4. Use the function 'C6X\_getvar' (help C6X\_getvar) in MATLAB to upload the LADAR's data into MATLAB. Then with the data in MATLAB create a x,y plot.
  - a. First set a break point at the closing brace of the while loop in `UART2ReceiveCharTask`.
  - b. Then in MATLAB find the number of data points the LADAR has collected. `Numreadings = C6X_getvar('numreadings_lastscan',1,'int32')`
  - c. Then upload the LADAR distance readings `Ladardist = C6X_getvar('LADARdistances',Numreadings);` and the LADAR angles, `Ladarangles = C6X_getvar('LADARangles',Numreadings);` into MATLAB. Then with the MATLAB 'pol2cart' function, convert the distance and angle readings to Cartesian X,Y coordinates. Make sure to scale the distance readings in millimeters to feet. Create an X,Y plot of the data.
  - d. Upload the LADAR X and Y data to compare to the X,Y plot you just created. `xdata = C6X_getvar('Xreflect',Numreadings);` and `ydata = C6X_getvar('Yreflect',Numreadings);`. Create an X,Y plot of the data and compare to your previous plot.
5. For this last section, we have created two M-files for you to work with. Both plot actual data from a robot's LADAR when inside the course. Use any hand calculations you choose to figure out the robot's location (x,y,theta) in the course. The two data sets are located in `v:\mechlab\lab7\LADAR1.m` and `v:\mechlab\lab7\LADAR2.m`. Just change the current MATLAB directory to `v:\mechlab\lab7` and type the name of the M-file to pull up the plots.

### Exercise 6: Grabbing the can

This is the point in the semester where you get to show off your mechanical creativity. Your task is to build a gripper for the robot so that it can grab an empty soda can in the upright position and take it to a known target and drop it off in the upright position. You are given a platform that you can modify however you desire. You may use up to two RC servos, three IR sensors and two feeler switches for this task. Your TA will show you the parts you may use to construct your

mechanism. Test this by placing an empty can directly in front of your robot. Drive the robot forward toward the can and, when close enough, grab the can.

When building this mechanism you should also look ahead to the last lab (final project). In the final project you are going to be asked to find and relocate the empty soda can, but you are also going to be collecting orange golf balls. These tasks do not need to be done at the same time but you should probably think about how you are going to handle the golf balls when designing the gripper for the soda can. For the final project you will be allowed more RC servos and IR sensors and you will be able to use the camera to find the can and golf balls if you wish.

### **Exercise 7: Finding and grabbing the can**

In the final project, a can will be placed within 2 feet of your robot. For this exercise, your robot must find the can, grab it and then let it go in the upright position 5 seconds later. Hint: Use the front IR sensor, but mount it so that it can sense the can *only* when the robot is aligned with the can. The IR sensor will also tell you when to 'grab'. When the program starts, swivel the robot around in place (velocity zero) until you find the can's direction, and then drive the robot forward slowly to grab it.

Modify your existing VB application to display your robot's position.

### **Lab Check Off:**

Show your TA your can-finding robot and its position updating on the VB display.